CLAIMS

What is claimed is:

1. A computer system, comprising:

a host processor, and

a network interface coupled to the host processor and to a network, the network interface comprising:

a first port that receives data from the host processor;

a second port that transmits data to the network;

a memory that stores data packets received by the first port, the memory being coupled to the first port and to the second port;

a control circuit that manages the memory as a plurality of queues having respective priorities, including logic to place a packet received from the host into one of the plurality of queues according to a quality of service parameter associated with the packet, and logic to service packets in the plurality of queues according to the respective priorities; and

logic to dynamically allocate space in said memory to the queues in the plurality of queues.

2. The computer system of claim 1, wherein the plurality of queues includes a higher priority queue, and a lower priority queue, and including a timeout timer coupled with the lower priority queue which is enabled if a packet is stored in the lower priority queue and expires after a timeout interval, and including logic to preempt the higher priority queue in favor of the lower priority queue if the timeout timer expires.

3. The computer system of claim 1, wherein the plurality of queues includes a higher priority queue, an intermediate priority queue, and a lower priority queue, and including

a first timeout timer coupled with the intermediate priority queue which is enabled if a packet is stored in the intermediate priority queue and expires after a first timeout interval, and including logic to preempt the higher priority queue in favor of the intermediate priority queue if the first timeout timer expires; and

a second timeout timer coupled with the lower priority queue which is enabled if a packet is stored in the lower priority queue and expires after a second timeout interval, and including logic to preempt the higher priority queue and the intermediate priority queue in favor of the lower priority queue if the second timeout timer expires.

4. The computer system of claim 1, wherein the plurality of queues includes a higher priority queue, an intermediate priority queue, and a lower priority queue, and including

an first timeout timer coupled with the intermediate priority queue which is enabled if a packet is stored in the intermediate priority queue and expires after a first timeout interval, and including logic to preempt the higher priority queue in favor of the intermediate priority queue if the first timeout timer expires;

a second timeout timer coupled with the lower priority queue which is enabled if a packet is stored in the lower priority queue and expires after a second timeout interval, and including logic to preempt the higher priority queue and the intermediate priority queue in favor of the lower priority queue if the second timeout timer expires; and

logic to service the intermediate priority queue in favor of the lower priority queue if both the first and second timeout timers expire.

5. The computer system of claim 1, further comprising logic in the network interface to execute a security process on packets in one of the plurality of queues.

6. The computer system of claim 1, wherein the second port further comprises circuitry for formatting packets according to a protocol compliant with an Ethernet protocol standard.

7. The computer system of claim 1, wherein the second port further comprises circuitry for formatting packets according to a protocol compliant with an InfiniBand protocol standard.

8. The computer system of claim 1, wherein the packets include frame start headers, and said quality of service parameters comprises codes in the frame start headers.

9. The computer system of claim 1, wherein said logic to dynamically allocate space in said memory maintains a list of free buffers and a list of used buffers for each of the plurality of queues.

10. The computer system of claim 1, wherein said memory comprises a single storage array.

11. The computer system of claim 1, wherein said logic to dynamically allocate space in said memory, places packets downloaded to the memory in a plurality of buffers in non-contiguous memory locations.

12. The computer system of claim 1, wherein said logic to dynamically allocate space in said memory maintains a list of free buffers and a list of used buffers for each of the plurality of queues, so that each virtual path has a free buffer list having a number of free buffers, and includes logic which releases a used buffer to the free buffer list for a queue in the plurality of queues having a smallest number of free buffers.

13. The computer system of claim 1, wherein said logic to dynamically allocate space in said memory maintains a list of free buffers and a list of used buffers for each of the plurality of queues, so that each virtual path has a free buffer list having a number of free buffers, and includes logic which releases a used buffer to the free buffer list for a queue in the plurality of queues having a largest amount of traffic.

14. The computer system of claim 1, wherein logic to dynamically allocate space in said memory maintains a list of buffer descriptors for corresponding buffers in said memory, the buffer descriptors including a parameter specifying a size of the corresponding buffer, and a location of the corresponding buffer.

15. The computer system of claim 1, wherein logic to dynamically allocate space in said memory maintains a list of buffer descriptors for corresponding buffers in said memory, the buffer descriptors including a programmable parameter specifying a size of the corresponding buffer.

16. The computer system of claim 1, wherein at least one queue in the plurality of queues comprises a first-in-first-out FIFO queue.

17. In a network interface apparatus, a method for managing transfer of data packets between a host processor and a network, comprising:

managing memory in the network interface apparatus as a plurality of queues having respective priorities, including placing a packet received from the host processor into one of the plurality of queues according to a quality of service parameter associated with the packet, and servicing packets in the plurality of queues according to the respective priorities; and

dynamically allocating space in said memory to the queues in the plurality of queues.

18. The method of claim 17, wherein the plurality of queues includes a higher priority queue, and a lower priority queue, and including enabling a timeout timer coupled with the lower priority queue if a packet is stored in the lower priority queue, the timeout timer expiring after a timeout interval, and preempting the higher priority queue in favor of the lower priority queue if the timeout timer expires.

19. The method of claim 17, wherein the plurality of queues includes a higher priority queue, an intermediate priority queue, and a lower priority queue, and including

enabling a first timeout timer coupled with the intermediate priority queue if a packet is stored in the intermediate priority queue, the first timeout timer expiring after a first timeout interval, and preempting the higher priority queue in favor of the intermediate priority queue if the first timeout timer expires; and

enabling a second timeout timer coupled with the lower priority queue if a packet is stored in the lower priority queue, the second timeout timer expiring after a second timeout interval, and preempting the higher priority queue and the intermediate priority queue in favor of the lower priority queue if the second timeout timer expires.

20. The method of claim 17, wherein the plurality of queues includes a higher priority queue, an intermediate priority queue, and a lower priority queue, and including

enabling a first timeout timer coupled with the intermediate priority queue if a packet is stored in the intermediate priority queue, the first timeout timer expiring after a first timeout

interval, and preempting the higher priority queue in favor of the intermediate priority queue if the first timeout timer expires; and

enabling a second timeout timer coupled with the lower priority queue if a packet is stored in the lower priority queue, the second timeout timer expiring after a second timeout interval, and preempting the higher priority queue and the intermediate priority queue in favor of the lower priority queue if the second timeout timer expires; and

servicing the intermediate priority queue in favor of the lower priority queue if both the first and second timeout timers expire.

21. The method of claim 17, further comprising executing a security process on packets in one of the plurality of queues.

22. The method of claim 17, including formatting packets in the network interface device according to a protocol compliant with an Ethernet protocol standard.

23. The method of claim 17, including formatting packets in the network interface device packets according to a protocol compliant with an Infiniband protocol standard.

24. The method of claim 17, wherein the packets include frame start headers, and said quality of service parameters comprises codes in the frame start headers.

25. The method of claim 17, wherein said dynamically allocating space in said memory includes maintaining a list of free buffers and a list of used buffers for each of the plurality of queues.

26. The method of claim 17, wherein said memory comprises a single storage array.

27. The method of claim 17, wherein said dynamically allocating space in said memory includes placing packets downloaded to the memory in a plurality of buffers in non-contiguous memory locations.

28. The method of claim 17, wherein said dynamically allocating space in said memory includes maintaining a list of free buffers and a list of used buffers for each of the plurality of queues, so that each virtual path has a free buffer list having a number of free buffers, and includes releasing a used buffer to the free buffer list for a queue in the plurality of queues having a smallest number of free buffers.

29. The method of claim 17, wherein said dynamically allocating space in said memory includes maintaining a list of free buffers and a list of used buffers for each of the plurality of queues, so that each virtual path has a free buffer list having a number of free buffers, and includes releasing a used buffer to the free buffer list for a queue in the plurality of queues having a largest amount of traffic.

30. The method of claim 17, wherein said dynamically allocating space in said memory includes maintaining a list of buffer descriptors for corresponding buffers in said memory, the buffer descriptors including a parameter specifying a size of the corresponding buffer, and a location of the corresponding buffer.

31. The method of claim 17, wherein said dynamically allocating space in said memory includes maintaining a list of buffer descriptors for corresponding buffers in said memory, the buffer descriptors including a programmable parameter specifying a size of the corresponding buffer.

32. The method of claim 17, including managing at least one queue in the plurality of queues as a first-in-first-out FIFO queue.

33. An integrated circuit for use in a network interface between a host processor and a network, comprising:

a first port that receives data from the host processor;

a second port that transmits data to the network;

a memory that stores data packets received by the first port, the memory being coupled to the first port and to the second port; and

a control circuit that manages the memory as a plurality of queues having respective priorities, including logic to place a packet received on the first port into one of the plurality of queues according to a quality of service parameter associated with the packet, and logic to transmit packets in the plurality of queues out the second port according to the respective priorities; and

logic to dynamically allocate space in said memory to the queues in the plurality of queues.

34. The integrated circuit of claim 33, wherein the plurality of queues includes a higher priority queue, and a lower priority queue, and including a timeout timer coupled with the lower priority queue which is enabled if a packet is stored in the lower priority queue and expires after a timeout interval, and including logic to preempt the higher priority queue in favor of the lower priority queue if the timeout timer expires.

35. The integrated circuit of claim 33, wherein the plurality of queues includes a higher priority queue, an intermediate priority queue, and a lower priority queue, and including

a first timeout timer coupled with the intermediate priority queue which is enabled if a packet is stored in the intermediate priority queue and expires after a first timeout interval, and including logic to preempt the higher priority queue in favor of the intermediate priority queue if the first timeout timer expires; and

a second timeout timer coupled with the lower priority queue which is enabled if a packet is stored in the lower priority queue and expires after a second timeout interval, and including logic to preempt the higher priority queue and the intermediate priority queue in favor of the lower priority queue if the second timeout timer expires.

36. The integrated circuit of claim 33, wherein the plurality of queues includes a higher priority queue, an intermediate priority queue, and a lower priority queue, and including

an first timeout timer coupled with the intermediate priority queue which is enabled if a packet is stored in the intermediate priority queue and expires after a first timeout interval, and including logic to preempt the higher priority queue in favor of the intermediate priority queue if the first timeout timer expires;

a second timeout timer coupled with the lower priority queue which is enabled if a packet is stored in the lower priority queue and expires after a second timeout interval, and including logic to preempt the higher priority queue and the intermediate priority queue in favor of the lower priority queue if the second timeout timer expires; and

logic to service the intermediate priority queue in favor of the lower priority queue if both the first and second timeout timers expire.

37. The integrated circuit of claim 33, further comprising logic in the network interface to execute a security process on packets in one of the plurality of queues.

38. The integrated circuit of claim 33, wherein the second port further comprises media access control circuitry for transmitting packets according to a protocol compliant with an Ethernet protocol standard.

39. The integrated circuit of claim 33, wherein the second port further comprises media access control circuitry for transmitting packets according to a protocol compliant with an Infiniband protocol standard.

40. The integrated circuit of claim 33, wherein the packets include frame start headers, and said quality of service parameters comprises codes in the frame start headers.

41. The integrated circuit of claim 33, wherein said logic to dynamically allocate space in said memory maintains a list of free buffers and a list of used buffers for each of the plurality of queues.

42. The integrated circuit of claim 33, wherein said memory comprises a single storage array.

43. The integrated circuit of claim 33, wherein said logic to dynamically allocate space in said memory, places packets downloaded to the memory in a plurality of buffers in non-contiguous memory locations.

44. The integrated circuit of claim 33, wherein said logic to dynamically allocate space in said memory maintains a list of free buffers and a list of used buffers for each of the plurality of queues, so that each virtual path has a free buffer list having a number of free buffers, and includes logic which releases a used buffer to the free buffer list for a queue in the plurality of queues having a smallest number of free buffers.

45. The integrated circuit of claim 33, wherein said logic to dynamically allocate space in said memory maintains a list of free buffers and a list of used buffers for each of the plurality of queues, so that each virtual path has a free buffer list having a number of free buffers, and includes logic which releases a used buffer to the free buffer list for a queue in the plurality of queues having a largest amount of traffic.

46. The integrated circuit of claim 33, wherein logic to dynamically allocate space in said memory maintains a list of buffer descriptors for corresponding buffers in said memory, the buffer descriptors including a parameter specifying a size of the corresponding buffer, and a location of the corresponding buffer.

47. The integrated circuit of claim 33, wherein logic to dynamically allocate space in said memory maintains a list of buffer descriptors for corresponding buffers in said memory, the buffer descriptors including a programmable parameter specifying a size of the corresponding buffer.

48. The integrated circuit of claim 33, wherein at least one queue in the plurality of queues comprises a first-in-first-out FIFO queue.